Interval scheduling

Input: set of intervals on the line, represented by pairs of points (ends of intervals)

Output: the largest set of intervals such that none two of them overlap

Dynamic Programming paradigm

Dynamic Programming (DP):

- Decompose the problem into series of sub-problems
- Build up correct solutions to larger and larger sub-problems

Similar to:

- Recursive programming vs. DP: in DP sub-problems may strongly overlap
- Exhaustive search vs. DP: in DP we try to find redundancies and reduce the space for searching

(Weighted) Interval scheduling

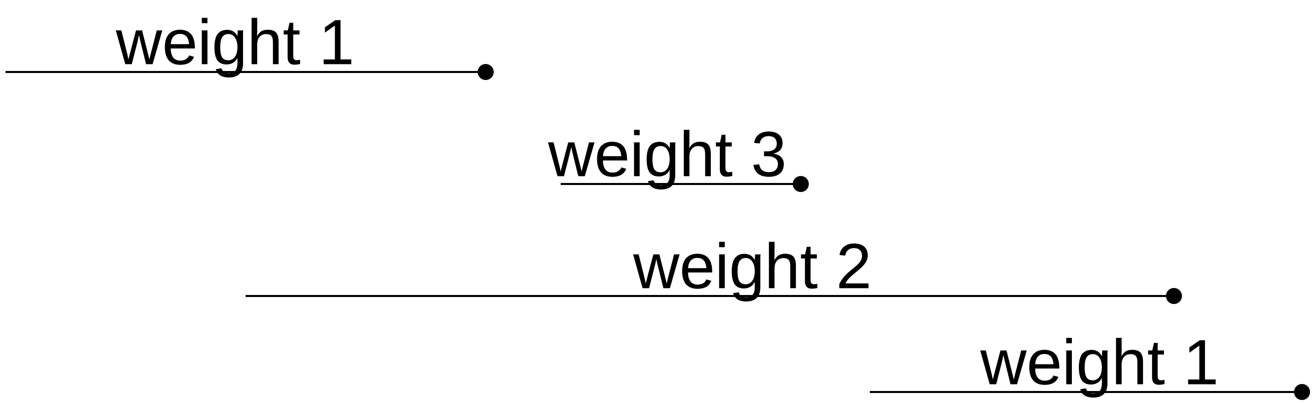
(Weighted) Interval scheduling:

Input: set of intervals (with weights) on the line, represented by pairs of points - ends of intervals

Output: the largest (maximum sum of weights) set of intervals such that none two of them overlap

Basic structure and definition

- Sort the intervals according to their right ends
- Define function p as follows:
  - $p(1) = 0$
  - $p(i)$ is the number of intervals which finish before i^{th} interval starts



$p(1)=0$

$p(2)=1$

$p(3)=0$

$p(4)=2$

Basic property

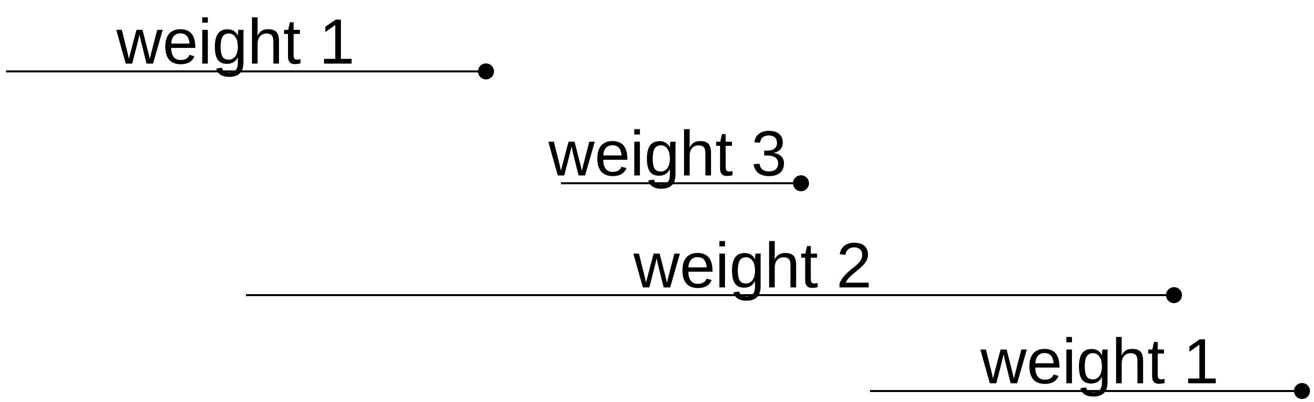
- Let w_j be the weight of j^{th} interval
- Optimal solution for the set of first j intervals satisfies

$$OPT(j) = \max\{\ w_j + OPT(p(j))\ ,\ OPT(j\text{-}1)\ \}$$

Proof:

If j^{th} interval is in the optimal solution **O** then the other intervals in **O** are among intervals $1,\ldots,p(j)$.

Otherwise search for solution among first j-1 intervals.



$p(1)=0$

$p(2)=1$

$p(3)=0$

$p(4)=2$

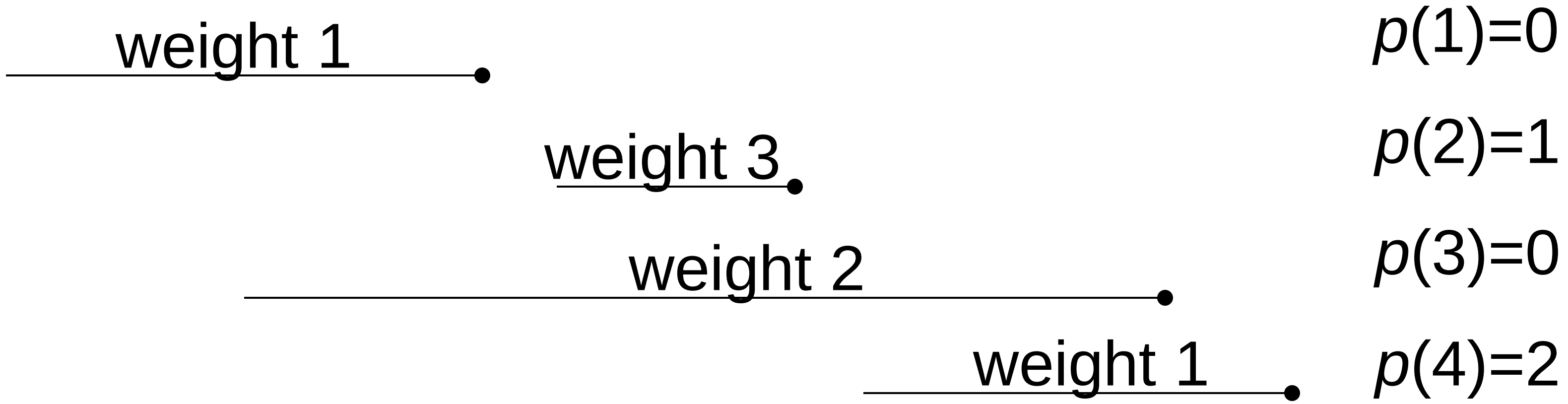
Sketch of the algorithm

- Additional array $M[0\ldots n]$ initialized by $0, p(1), \ldots, p(n)$

(intuitively $M[j]$ stores optimal solution $OPT(j)$)

Algorithm

- For $j = 1, \ldots, n$ do
  - Read $p(j) = M[j]$
  - Set $M[j] := \max\{ w_j + M[p(j)] , M[j-1] \}$



Consider the set of weighted intervals given below, where s_i is the start time, f_i is the finish time, and v_i is the value of the interval.

j	s_j	f_j	v_j	$p(j)$
1	0	6	2	0
2	2	10	4	0
3	9	15	6	1
4	7	18	7	1

Solve this instance of the weighted interval scheduling problem, i.e. find a set of (non-conflicting) intervals with maximum total weight.

$$Opt(j) = \max\left\{Opt(j-1), v_j + Opt(p(j))\right\}.$$

Therefore, we have

$$\begin{aligned}
Opt(0) &= 0 \\
Opt(1) &= \max\left\{Opt(0),\ v_1 + Opt(p(1))\right\} = 2 \\
Opt(2) &= \max\left\{Opt(1),\ v_2 + Opt(p(2))\right\} = \max\{2,\ 4+0\} = 4 \\
Opt(3) &= \max\left\{Opt(2),\ v_3 + Opt(p(3))\right\} = \max\{4,\ 6+2\} = 8 \\
Opt(4) &= \max\left\{Opt(3),\ v_4 + Opt(p(4))\right\} = \max\{8,\ 7+2\} = 9
\end{aligned}$$

Complexity of solution

Time: $O(n \log n)$

- Sorting: $O(n \log n)$
- Initialization of $M[0\ldots n]$ by $0, p(1), \ldots, p(n)$: $O(n \log n)$
- Algorithm: n iterations, each takes constant time, total $O(n)$

Memory: $O(n)$ - additional array M